Developer & Work Shop

Visual FoxPro

: รู้จัก ActiveX

Visual Basic

: Visual Basic กับ DirectX

โยธิน อินบรรเลง

รู้จัก ActiveX Control ตอนที่ 1

ช่วยทำให้แต่ละโปรแกรมสามารถปฏิบัติงานร่วมกับแอพพลิเคชันอื่น ๆ ภายใต้สภาพแวดล้อมเดียวกัน

Visual FoxPro

สวัสดีครับ พบกันอีกครั้งหลังจากห่างหายกันไปนาน แต่ใจนั้นยังวนเวียนอยู่กับ FoxPro ไม่ไปไหน ในช่วงเวลาที่ผ่านมาที่หายไปก็ได้ศึกษาถึงรูปแบบใหม่เพื่อนำเสนอยังท่านผู้อ่านให้ได้แนวทางแปลกใหม่เช่นเคย ฉบับนี้เช่นกัน ผมได้นำเอาเรื่องราวเกี่ยวกับ ActiveX หรือเดิมที่เป็น OLE (Object Linking Embedded) โดยจะนำเอาคอนโทรลที่เป็น ActiveX ทั้งของ Microsoft และไม่ใช่มาสาธยายกันซักที

ที่สำคัญมากกว่านั้นคือเหล่าบรรดาโปรแกรมเมอร์ Visual FoxPro จะได้หายสงสัยว่าโปรแกรมที่กำลังใช้พัฒนากำลังจะล้าหลังหรือเปล่าหรืออาจจะสงสัยประสิทธิภาพจะสู้คอมไพเลอร์อื่น ๆ ได้หรือไม่

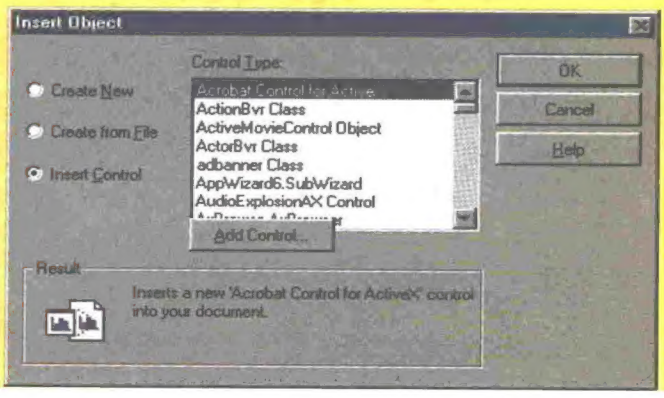
รู้จัก ActiveX

ActiveX เป็นเทคโนโลยีของไมโครซอฟท์ที่ทำให้แต่ละโปรแกรมสามารถปฏิบัติงานร่วมกับแอพพลิเคชันอื่น ๆ ภายใต้สภาพแวดล้อมเดียวกัน เช่น โปรแกรม Visual FoxPro สามารถใช้งานส่วนประกอบ (Component Object Model, COM) ที่สร้างจาก Visual C++ หรือ Delphi ได้ แต่ใช่ว่าทุกโปรแกรมจะสามารถใช้งาน COM ได้เสมอไป โปรแกรมที่จะสามารถใช้งานข้ามระบบได้จะต้องเป็นโปรแกรมที่มีคุณสมบัติสนับสนุนการทำงานแบบ OLE ด้วย

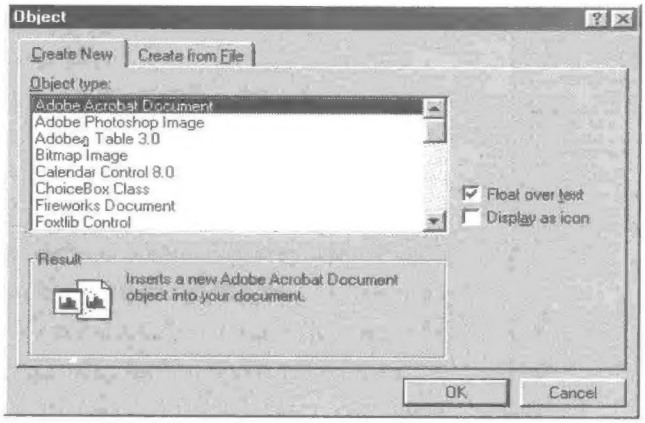
การเรียกใช้งาน ActiveX ผ่าน Visual FoxPro

วิธีการเรียกใช้ ActiveX ผ่าน Visual FoxPro จะสามารถทำได้โดยแทรกคอนโทรลแบบ ActiveX เข้าไปในฟอร์ม โดยทำได้ 2 วิธีคือ

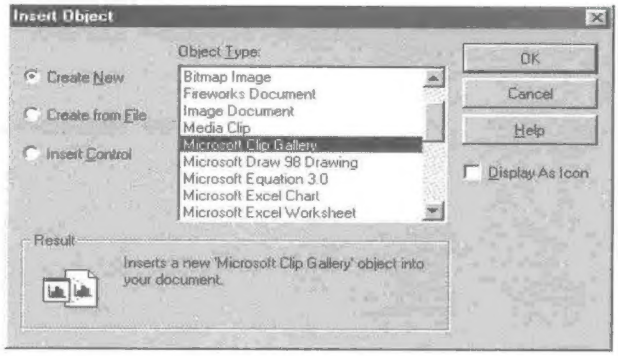
1. จัดการคอนโทรลผ่าน Form Control ของ Visual FoxPro
2. แทรก ActiveX ผ่านคอนโทรล OLE (ActiveX Control)



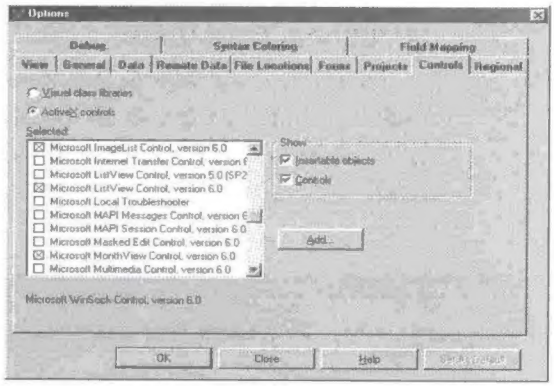
รูปที่ 1 การแทรก Object จาก Visual FoxPro ซึ่งสนับสนุนการใช้งาน Activex ได้



รูปที่ 2 การแทรก Object จาก MS-WORD ซึ่งไม่สามารถเพิ่มคอนโทรล ActiveX ได้โดยตรง



รูปที่ 4 หลังจากเลือก OLE Control แล้วจะมีกรอบสำหรับแทรก Object ให้เลือกคลิกที่ Insert Control (สำหรับบางเครื่องอาจจะใช้เวลานาน) จากนั้นให้เลือกคอนโทรลภายในช่อง Control Type เมื่อได้คอนโทรลที่ต้องการแล้วคลิก OK เพื่อกลับไปยังฟอร์มตามเดิม



รูปที่ 3 เลือกเมนู Options, เลือกแท็บ Form และเลือก ActiveX Control ผู้ใช้สามารถที่เลือก Control ชนิด ActiveX มาไว้ใน Form Control เมื่อเลือกเสร็จให้คลิก Set As Default

จาก Form Control ให้เลือกคลิก View Class เพื่อเลือกชุดคอนโทรลชนิด ActiveX

1. จัดการคอนโทรลผ่าน Form Control

วิธีการจัดการคอนโทรลผ่าน Form Control จะมีขั้นตอนมากมายไปสักนิด แต่ว่าเราก็สามารถเลือกใช้คอนโทรลได้สะดวกและรวดเร็ว

2. แทรก ActiveX ผ่านคอนโทรล OLE (ActiveX Control)

วิธีการแทรก ActiveX ผ่านคอนโทรล OLE จะทำได้ง่ายเพียงคลิกคอนโทรล OLE Control (ActiveX Control) จาก Form Control แล้วนำมาวางที่ฟอร์ม หลังจากนั้นก็เลือกชนิด ActiveX ที่ต้องการก็สามารถนำมาใช้งานได้เลย

เราได้ทราบวิธีการเรียกใช้ คอนโทรลชนิด ActiveX เรียบร้อยต่อไปจะเป็นการแนะนำและตัวอย่างการเขียนโปรแกรมร่วมกับ ActiveX โดยจะเริ่มที่ออบเจ็กต์แรก คือ Microsoft Clip Gallery

1. Microsoft Clip Gallery

หลายท่านที่คุ้นเคยกับ MS-Word จะทราบดีว่าจะมี Clip Art สวย ๆ เยอะแยะ โดยเฉพาะไฟล์ชนิด *.wmf (Windows Meta File) ซึ่งเป็นภาพลายเส้นไม่สามารถนำมาใช้งานกับ Visual FoxPro โดยตรง แต่เราสามารถที่จะนำมาไว้ในฟอร์มผ่าน ActiveX คอนโทรลได้

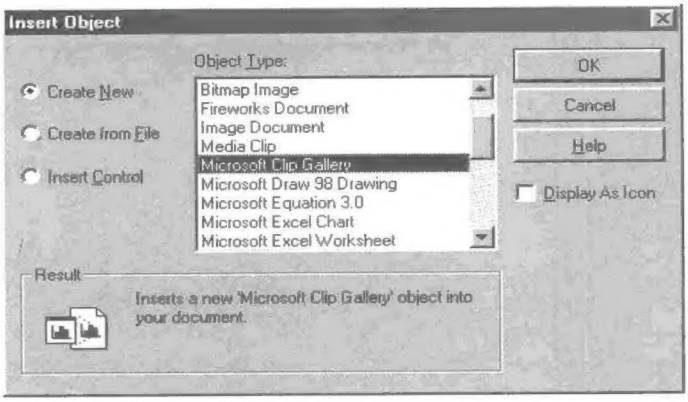
วิธีการไม่ยุ่งยากเพียงแต่เปิดฟอร์มแล้วเลือกคอนโทรล Microsoft Clip Gallery แล้วเลือกรูปภาพก็ใช้ได้ทันที

Property ที่ต้องกำหนด

Stretch 0 - Clip (Default) กำหนดให้รูปภาพแสดงตามขนาดคอนโทรลที่เลือกถ้าภาพใหญ่กว่าจะถูกตัดออกไป

1 - Isometric กำหนดให้รูปภาพมีสัดส่วนเหมือนเดิมแต่จะจัดขนาดให้พอดีกับขนาดของคอนโทรล

2 - Stretch กำหนดให้จัดขนาดรูปภาพตามขนาดของคอนโทรลซึ่งจะไม่สนใจสัดส่วนของภาพ



รูปที่ 5 เลือกคอนโทรล OLE แล้วเลือก Create New / Microsoft Clip Gallery

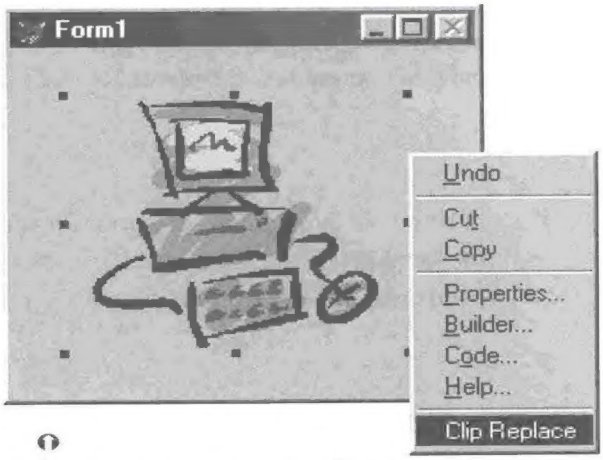
2. Microsoft TreeView

คอนโทรลชนิด OLE ในรายการที่ผ่านมาเป็นเพียงรูปแบบการใช้งานที่ง่ายไม่ซับซ้อนเพียงแค่กำหนดคุณสมบัติบางประการก็สามารถใช้งานได้ แต่จริง ๆ แล้ว ActiveX อีกหลายชนิดที่ต้องกำหนดคำสั่งก่อนถึงจะทำงานได้สมบูรณ์แบบ ในหัวข้อนี้จะได้กล่าวถึง ActiveX ที่ต้องเขียนคำสั่งควบคุมด้วยซึ่งมีความซับซ้อนพอสมควร แต่ถ้าท่านผู้อ่านค่อยพิจารณาทำความเข้าใจหลักการแล้วจะพบว่าไม่มีอะไรยากเกินกว่าความตั้งใจของเรา

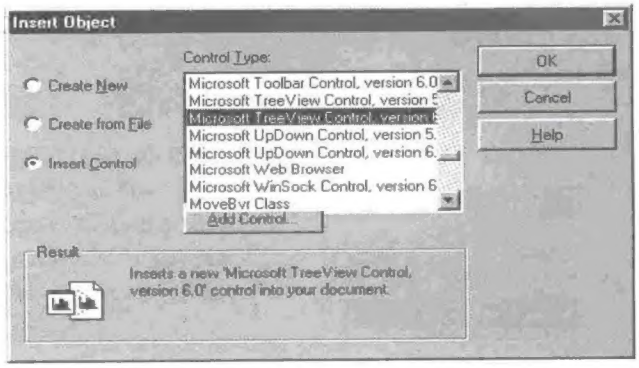
Microsoft TreeView เป็นคอนโทรลสำหรับแสดงข้อมูลแบบโครงสร้างต้นไม้ (Tree View) ซึ่งเราสามารถนำข้อมูลมาแสดงได้ตามต้องการพร้อมกับสามารถกำหนดรูปแบบให้สวยงามด้วยการเพิ่มรูปภาพลงไปอีกด้วย

รูปแบบของ Tree View

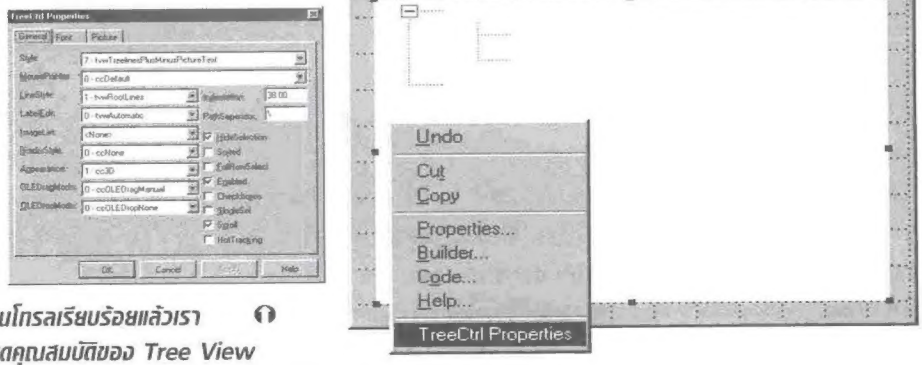
ตัวอย่างของโปรแกรมใช้งาน Tree View ที่เราคุ้นเคยกันดีก็คือ Windows Explorer นั่นเองแต่ Tree View ไม่ได้ใช้สำหรับแสดงชื่อไฟล์แบบ Windows Explorer เท่านั้นแต่ยังสามารถแสดงข้อมูลอื่น ๆ อีกหลายประการ



รูปที่ 6 หลังจากนำ Clip Art มาวางในฟอร์มเรียบร้อย จะสามารถเปลี่ยนรูปภาพโดยคลิกเมาส์ขวาที่คอนโทรล แล้วเลือก Clip Replace



รูปที่ 7 เลือกคอนโทรล OLE แล้วเลือก Insert Control/Microsoft TreeView Control Version 6.0



รูปที่ 8 เมื่อเลือกคอนโทรลเรียบร้อยแล้วเรา สามารถกำหนดคุณสมบัติของ Tree View โดยเลือกคลิกเมาส์ขวา และเลือก TreeCtrl Properties

คุณสมบัติที่ควรรู้จัก

Style สำหรับกำหนดรูปแบบที่จะกำหนดให้กับ TreeView

รูปแบบการใช้งาน

object.Style [= number]

number คือหมายเลขรูปแบบที่จะกำหนดมีดังนี้

0 แสดงเฉพาะข้อความ
1 แสดงข้อความและรูปภาพ
2 แสดงสัญลักษณ์ + / - และข้อความ
3 แสดงสัญลักษณ์ + / - ข้อความ และ รูปภาพ
4 แสดงเส้นและข้อความ
5 แสดงเส้น ข้อความ และรูปภาพ
6 แสดงเส้น สัญลักษณ์ +/- และข้อความ
7 แสดงเส้น สัญลักษณ์ +/- รูปภาพและข้อความ (ค่าปกติ)

LineStyle สำหรับกำหนดรูปแบบของเส้นระหว่าง Node (หัวข้อหลัก)

รูปแบบการใช้งาน

object.**LineStyle** [= number]

number คือหมายเลขที่จะกำหนดรูปแบบเส้นระหว่าง Node (หัวข้อหลัก) กำหนดได้ดังนี้

0 - กำหนดให้แสดงเส้นแบบ Tree Lines ซึ่งจะไม่แสดงเส้นระหว่าง Node (tvwTreeLines)

1 - กำหนดให้แสดงเส้นแบบ Root Lines ซึ่งจะแสดงเส้นระหว่างแต่ละหัวข้อ (tvwRootLines)

LabelEdit สำหรับกับรูปแบบการแก้ไขข้อความที่แสดงใน Tree View

รูปแบบการใช้งาน

object.LabelEdit [= integer]

integer คือหมายเลขที่จะกำหนดรูปแบบการแก้ไขข้อความ สามารถกำหนดได้ดังนี้

0 - tvwAutomatic กำหนดให้สามารถแก้ไขข้อความได้ เมื่อคลิกเมาส์ที่ข้อความ (ปกติ)

1 - tvwManual กำหนดให้แก้ไขข้อความผ่านการเขียนโปรแกรมเท่านั้น

ImageList กำหนดชื่อคอนโทรลที่บรรจุรูปภาพที่จะแสดงใน TreeView

รูปแบบการใช้งาน

object.ImageList = ControlName

ControlName คือ ชื่อของคอนโทรล ImageList ที่บรรจุรูปภาพ

HideSelection กำหนดให้แสดงหรือไม่แสดงแถบแสงเลือกข้อความ เมื่อโฟกัสออกไปนอก TreeView

รูปแบบการใช้งาน

object.HideSelection = lExpression

lExpression คือ ตรรกะที่ใช้กำหนดให้ซ่อนแถบแสงเลือกข้อความ .T. - ซ่อน.F. แสดง

เมธอดสำหรับควบคุมการทำงาน TreeView

เนื่องจาก TreeView จะแบ่งออกเป็น Object ซึ่งการเขียนโปรแกรมจะต้องอิงกับ Object ด้วยดังนั้น TreeView จึงมีเมธ็อดสำหรับเรียกใช้งานน้อย

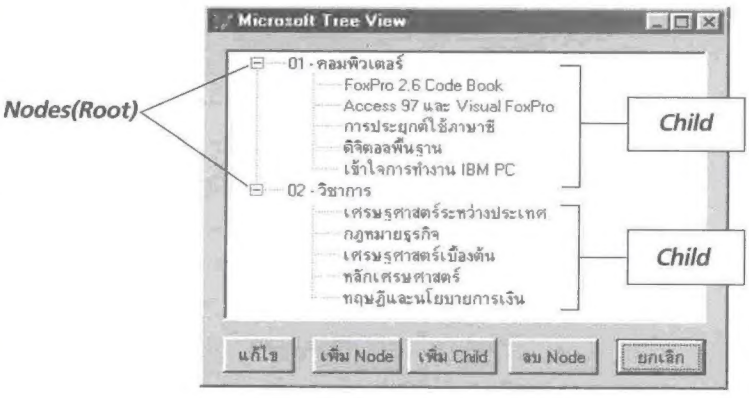
StartLabelEdit สำหรับแก้ไขข้อความที่เลือกกรณีที่กำหนดให้ LabelEdit = 1 (tvwManual)

รูปแบบการใช้งาน

object.StartLabelEdit หลังจากที่เมธอดนี้ถูกเรียกใช้แล้วข้อความที่เลือกจะสามารถเปลี่ยนแปลงได้

Object ใน TreeView

ภายในคอนโทรล TreeView จะประกอบไปด้วย Objects ย่อยอีก คือ Nodes (Nodes Collection) จะเก็บรายละเอียดและกำหนดการทำงานของ Tree View อีกระดับหนึ่ง การอ้างถึงก็จะเหมือนกับการอ้างถึงคอนโทรลอื่น ๆ และ Nodes จะประกอบด้วยพร็อปเพอร์ตี้และเมธอด



รูปที่ 9
แสดงรูปแบบของความสัมพันธ์ระหว่าง Root,Node และ Child

การเพิ่มข้อมูลใน TreeView จะมีอยู่ 2 แบบคือเพิ่ม Nodes ซึ่งเป็นรายการหลักหรือรายการนอกสุดที่เรียกว่า Node ซึ่งจะอยู่ในระดับ Root และการเพิ่ม Child คือ รายการย่อย (Node Child) ในแต่ละหัวข้อ

Nodes Collection

Nodes เป็นออปเจ็กต์หนึ่งที่บรรจุอยู่ภายใน Tree View ซึ่งประกอบไปด้วย พร็อปเพอร์ตี้ และเมธ็อดหลายรายการ แต่ในบทความนี้จะกล่าวถึงเฉพาะรายการที่จำเป็นต้องใช้งานเท่านั้นเพราะบางรายการไม่ได้เรียกใช้งาน

พร็อปเพอร์ตี้ที่ควรรู้จัก	
พร็อปเพอร์ตี้	**รายละเอียด**
Bold	กำหนดรูปแบบตัวอักษรตัวหนาให้กับข้อความของ Node (.T. = หนา, .F. = ธรรมดา)
Checked	กำหนดหรือตรวจสอบการคลิกปุ่ม Check กรณีมีการใช้ CheckBox ใน TreeView (.T. = ถ้า CheckBox ถูกกาเลือก, .F. ถ้า Check Box ไม่ถูกเลือก)
Children	ให้ค่าจำนวน Child ที่อยู่ภายใต้แต่ละ Node
Expanded	ส่งสถานะการย่อหรือขยาย Nodes กลับคืน (.T. = ขยาย .F. = ย่อ)
ExpanedImage	กำหนดรูปภาพที่จะแสดงหน้าข้อความเมื่อขยาย Node (ดูจากตัวอย่างการเพิ่มรูปภาพ)
FullPath	ส่งชื่อเต็มทั้งคีย์และข้อความของแต่ละ Node
Selected	แสดงสถานะการเลือกคลิก Node (.T. = เลือก .F. = ไม่เลือก)
SelectedImage	กำหนดรูปภาพให้กับ Node ที่ถูกเลือก (ดูจากตัวอย่างการเพิ่มรูปภาพ)

เมธอดสำหรับ Node

เราทราบแล้วว่า Node เป็นออปเจ็กต์ หนึ่งซึ่งประกอบด้วย พร็อปเพอร์ตี้ และเมธอดดังได้อธิบายถึงพร็อปเพอร์ตี้ที่สำคัญไปแล้ว ต่อไปนี้จะกล่าวถึงเมธอดสำคัญ 2 เมธอดคือ Add และ Remove

ระหว่าง 2 เมธอดนี้จะพบปัญหาขณะเขียนโปรแกรมคือ เมธอด Add จะหาอ่าน Help ยากเล็กน้อยส่วน Remove จะไม่มีตัวอย่างหรือรายละเอียดเลย ทำให้การเขียนต้องลองผิดลองถูกเปรียบเทียบกับการลบข้อมูลจาก List Box

เมธอดที่เกี่ยวข้อง	
เมธอด	**รายละเอียด**
Add	**เมธอดสำหรับเพิ่ม Node หรือ Child ให้กับ Tree View**
พารามิเตอร์	
relative	หมายเลขคีย์ของ Node ที่เราจะเพิ่ม Child เข้าไปอีก ถ้าไม่กำหนดจะเป็นการเพิ่มที่ Root และการเพิ่มจะถูกกำหนดรูปแบบโดยพารามิเตอร์ Relationship (จะระบุหรือไม่ก็ได้)
Relationship	กำหนดความสัมพันธ์ระหว่าง Node หรือ Child ที่เพิ่มกับ Node ที่เลือก (ดูรายละเอียดในกรอบตัวเลขกำหนดความสัมพันธ์)
key	คีย์ของแต่ละ Node จะต้องเป็นข้อมูลชนิดตัวอักษรเท่านั้นและถ้าจำเป็นต้องใช้ตัวเลขขึ้นต้นให้ใช้ สัญลักษณ์พิเศษ เช่น_หรือ ใช้ตัวอักษร A-Z ขึ้นต้นด้วย
text	ข้อความของ Node ที่เพิ่ม
Image	ตัวชี้นำในการเลือกรูปภาพจะเป็นหมายเลขหรือข้อความที่เป็น Key (ดูจาก Image List)
Selectedimage	ตัวชี้นำสำหรับรูปภาพของ Node ที่ถูกเลือกจะเป็นหมายเลขหรือข้อความที่เป็น Key (ดูจาก Image List)
Remove	ลบรายการออกจาก TreeView พารามิเตอร์
พารามิเตอร์	
keyDelete	ชื่อคีย์สำหรับอ้างอิงในการลบ

อีเวนต์ใน TreeView

อีเวนต์คือเหตุการณ์ที่เกิดขึ้นกับคอนโทรล TreeView เหตุการณ์ที่เกิดขึ้นจะเป็นอะไรก็ได้ที่โปรแกรมกำหนด เช่น การกดปุ่มคีย์บอร์ด (Keypress) หรือ การคลิกเมาส์ (MouseClick) ซึ่งในคอนโทรล TreeView นี้จะมีเหตุการณ์ที่สำคัญคือ

เพิ่มรูปให้กับ TreeView

จากโปรแกรมตัวอย่างที่ผ่านมาจะเพิ่มข้อความให้ กับ Node และ Child ทางด้านหน้าข้อความจะเป็นเพียงเครื่องหมาย + หรือ - เมื่อมีการคลิกเลือก Node เท่านั้น ถ้าสังเกตในโปรแกรม Windows Explorer จะเห็นว่ามีรูปภาพอยู่ด้านหน้า ข้อความที่แสดงด้วย

ในโปรแกรมของเราที่ประกอบด้วย Tree View ก็สามารถนำรูปภาพมาใช้ได้เหมือนกันโดยผ่าน คอนโทรลชนิด ActvieX ชื่อ ImageList ซึ่งมีรายละเอียดดังนี้

อีเวนต์สำคัญของ TreeView	
อีเวนต์	**รายละเอียด**
Expand	เหตุการณ์เมื่อผู้ใช้ขยาย Node
NodeClick	เหตุการณ์เมื่อผู้ใช้คลิกที่ Node
NodeCheck	เหตุการณ์เมื่อผู้ใช้คลิกกา CheckBox
Double Click	เหตุการณ์เมื่อผู้ใช้เมาส์ดับเบิลคลิกบริเวณ Tree View (เหมือนกับคอนโทรลทั่วไป)

3. Microsoft Image List Control

คอนโทรล Image List ของไมโครซอฟท์จะให้ผู้ใช้กำหนดรูปภาพสำหรับใช้แสดงในคอนโทรลอื่น ๆ ที่เกี่ยวข้องคือ Microsoft Tree View, Microsoft OutLine, และ Microsoft List View เป็นต้น การใช้งานจะต้องกำหนดรูปภาพเสียก่อนดังนี้

1. การกำหนดรูปภาพที่จะใช้งาน

การกำหนดรูปภาพจะเริ่มจากนำคอนโทรล Image List มาไว้ในฟอร์มก่อนแล้วคลิกขวาที่คอนโทรลแล้วเลือกคลิกปุ่มขวาของเมาส์เพื่อกำหนดพร็อปเพอร์ตี้ของคอนโทรลจะได้ดังรูปภาพ

2. วิธีการใช้งาน

เมื่อกำหนดรูปภาพเรียบร้อยแล้วให้ตั้งชื่อคอนโทรลนี้เป็น img จากนั้นก็จะเป็นวิธีการนำเอารูปภาพที่อยู่ใน ImageList ไปใช้งาน ให้กำหนดในอีเวนต์ Init ของคอนโทรลที่เป็น TreeView ในที่นี้กำหนดชื่อเป็น Tree ดังนี้

อีเวนต์ Tree.Init

```
This.ImageList = ThisForm.Img    && กำหนดให้อ้างอิงถึงรูปภาพตามลำดับ
ในคอนโทรลชื่อ img
```

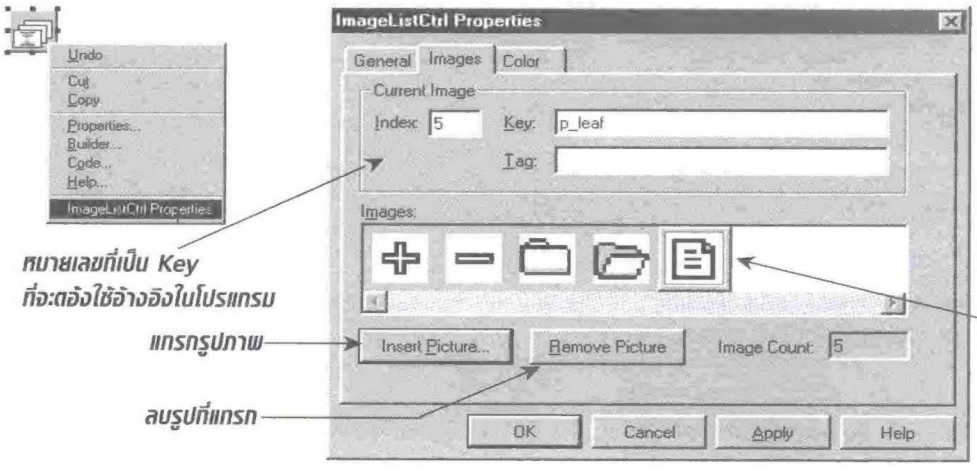
และการอ้างอิงรูปภาพทำได้ดังนี้

```
rNode = .Nodes.Add(,,cRoot,cGrp)
rNode.Image = 3
```

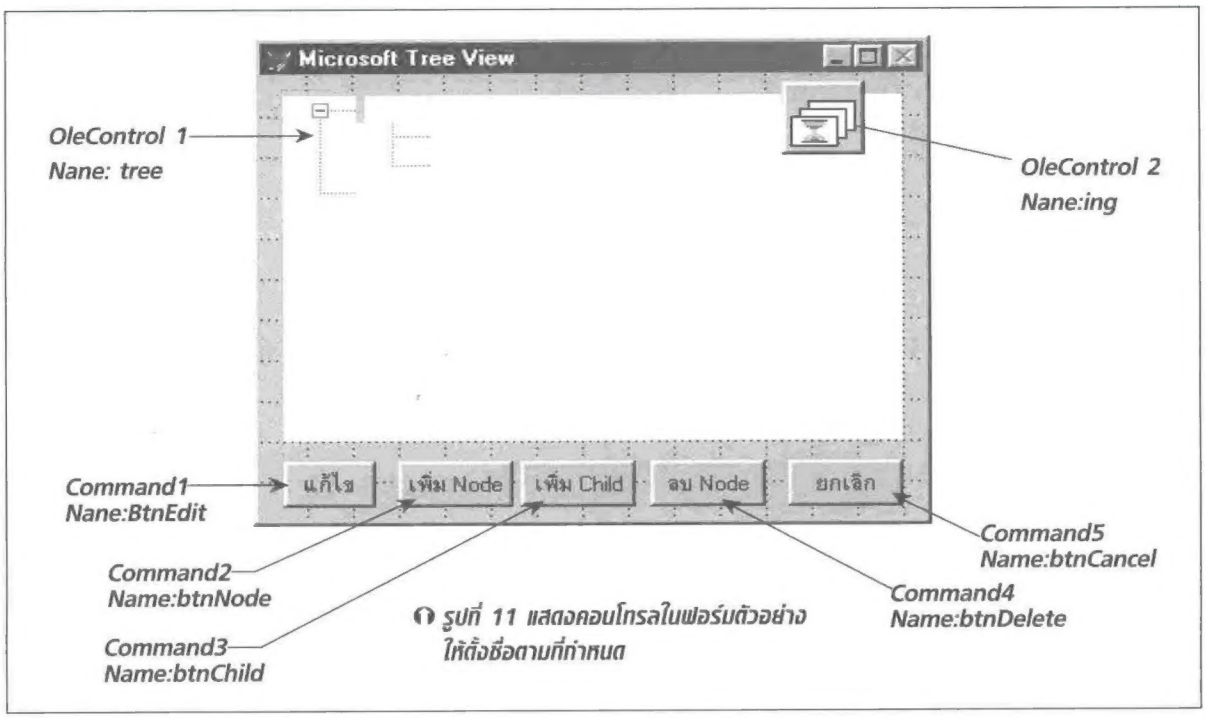
จากตัวอย่าง เมื่อเพิ่ม Node ด้วยเมธอด Add แล้วโปรแกรมจะอ้างอิงกับตัวแปรที่กำหนดทางด้านซ้ายมือคือ rNode และการอ้างอิงรูปภาพจะสามารถกำหนดหมายเลข Index Key ได้โดยตรงในตัวอย่างนี้กำหนดให้แสดงรูปหมายเลข 3

ตัวอย่างการเขียนโปรแกรมควบคุม TreeView

ตัวอย่างต่อไปนี้จะแสดงวิธีการเขียนโปรแกรมควบคุมการทำงาน TreeView ซึ่งตัวอย่างนี้จะเขียนแบบง่าย ๆ ไม่ซับซ้อนมากนักและมีคำอธิบายในบรรทัดที่สำคัญ ก่อนอื่นท่านผู้อ่านต้องสร้างฟอร์มพร้อมกำหนดคุณสมบัติของฟอร์มและคอนโทรลตามในรูปเสียก่อน



รูปที่ 10 เมื่อคลิกเมาส์ปุ่มขวาและเลือก ImageListCtrl Properties



รูปที่ 11 แสดงคอนโทรลในฟอร์มตัวอย่าง ให้ตั้งชื่อตามที่กำหนด

```
Control: 1 - Form1
Property
        ckey = 0    && สร้างขึ้นใหม่จากเมนู Form/New Property
Events: Init
PUBLIC arrGrp,arrBook              && ประกาศให้อะเรย์ arrGrp, arrBook สามารถใช้ได้ทุกโมดูล
DECLARE arrGrp(2), arrBook(2,5)    && กำหนดขนาดอะเรย์
        * กำหนดค่าให้กับ arrGrp ซึ่งเป็นหัวข้อหรือใช้แสดงในส่วนของ Node ในตัวอย่างนี้มีเพียง 2 Node
        arrGrp(1) = '01 - คอมพิวเตอร์'
        arrGrp(2) = '02 - วิชาการ'
        * กำหนดค่าให้กับ arrBook ซึ่งเป็นหัวข้อย่อยในแต่ละ Node ในตัวอย่างนี้จะมี 5 รายการต่อ Nodes
        arrBook(1,1) = 'FoxPro 2.6 Code Book'
        arrBook(1,2) = 'Access 97 และ Visual FoxPro'
        arrBook(1,3) = 'การประยุกต์ใช้ภาษาซี'
        arrBook(1,4) = 'ดิจิตอลพื้นฐาน'
        arrBook(1,5) = 'เข้าใจการทำงาน IBM PC'
        arrBook(2,1) = 'เศรษฐศาสตร์ระหว่างประเทศ'
        arrBook(2,2) = 'กฎหมายธุรกิจ'
        arrBook(2,3) = 'เศรษฐศาสตร์เบื้องต้น'
        arrBook(2,4) = 'หลักเศรษศาสตร์'
        arrBook(2,5) = 'ทฤษฎีและนโยบายการเงิน'
        ThisForm.DspTree           && แสดงข้อมูลในอะเรย์ลงใน TreeView
Events : dsptree
        LOCAL i,r   && ประกาศให้ตัวแปร i และ r เป็นตัวแปรที่ใช้เฉพาะในเมธอด DspTree เท่านั้น
        tvwChild = 4&& กำหนดรูปแบบความสัมพันธ์ระหว่าง Node (4 = กำหนดให้เป็น Node ลูก)
        WITH ThisForm.Tree
                r = 1
                .Nodes.Clear       && เรียกเมธอด Clear ของ Node เพื่อลบข้อมูลเดิมใน TreeView
```

```
                nGrpLen = ALEN(arrGrp)&& กำหนดจำนวน Node
                nSubLen = ALEN(arrBook)/nGrpLen   && กำหนดจำนวน Node ย่อย
                FOR iRoot = 1 TO nGrpLen
                cRoot = 'R'+ALLTRIM(STR(iRoot))    && กำหนด Key ที่จะใช้อ้างอิงใน Node
                cGrp = arrGrp(iRoot)      && ข้อความที่จะแสดงใน Node หลัก นำมาจาก arrGrp
                rNode = .Nodes.Add(,,cRoot,cGrp)    && เพิ่ม Nodes เข้าไปยัง TreeView
                *** เพิ่ม Nodes ย่อยเข้าไปใน Node หลัก
                ThisForm.cKey = iRoot
                FOR iChild = 1 TO nSubLen
                        * กำหนด Key ให้กับ Node ย่อย
                        cRootSub = 'S'+ALLTRIM(STR(iRoot))+ALLTRIM(STR(iChild))
                        * เพิ่ม Nodes ย่อยให้กับ Nodes หลักโดยอ้างอิงไปยัง key ที่กำหนดไว้ใน cRoot
                        oSubNode = .Nodes.Add(cRoot, tvwChild, cRootSub, arrBook(iRoot,iChild))
                        * กำหนดสีให้กับ Node ย่อย
                                oSubNode.ForeColor = RGB(0,0,192)
                        NEXT iChild
                NEXT iRoot
        ENDWITH

Control: 2 - tree
Events : Init
        * กำหนดคอนโทรลที่บรรจุรูปภาพ (จะต้องกำหนดด้วยคำสั่งเท่านั้นไม่สามารถจะกำหนดผ่านพร็อปเพอร์ตี้ขณะ
        * สร้างฟอร์มโดยตรง)
        This.ImageList = ThisForm.Img

Events : Collapse
        * เมื่อ Node ถูกย่อลงอีเวนต์ Collapse จะถูกเรียกใช้
        *** ActiveX Control Event ***
        LPARAMETERS node
        IF !This.Nodes(node.index).Expanded
                This.Nodes(node.index).Image = 3
        ENDIF

Events : Expand
        *** ActiveX Control Event ***
        LPARAMETERS node
        IF This.Nodes(node.index).Expanded
                This.Nodes(node.index).Image = 4
        ENDIF

Events : DblClick
        *** ActiveX Control Event ***
        WITH This
                nIdx = This.SelectedItem.Index      && เก็บค่าหมายเลขชี้นำ (Index)
                cKey = This.Nodes(nIdx).Key         && เก็บค่าคีย์โดยระบุจากตัวชี้นำ
                IF LEFT(cKey,1) = 'R'               && ถ้าค่าของคีย์เป็น 'R' แสดงว่าเป็น Node หลักให้ย้อนกลับ
                RETURN
                ELSE      && แต่ถ้าไม่ใช่จะแสดงหมายเลขชี้นำ, คีย์, ข้อความ และ จำนวน Chile
```

```
                    *** ตรงนี้ผู้ใช้สามารถประยุกต์คำสั่งอย่างอื่นได้ตามต้องการ ***
                    cText = .Nodes(nIdx).Text
                    cKey = .Nodes(nIdx).Key
                    cIndex = ALLTRIM(STR(nIdx))
                    cChild = ALLTRIM(STR(.Nodes(nIdx).Children))
                    =MESSAGEBOX('Index = '+cIndex+CHR(13)+;
                            'Key = '+cKey+CHR(13)+;
                            'Text = '+cText+CHR(13)+ ;
                            'Child(s) = '+cChild, ;
                            0+64+0,'You are selected')
                EDIF
        ENDWITH
Control: 3 - btnEdit
Events: Click
* เมื่อคลิกปุ่ม แก้ไข โปรแกรมจะเรียกเมธ็อด StartLabelEdit เพื่อแก้ไขข้อความ
        ThisForm.Tree.StartLabelEdit

Control: 4 - btnNode
Events: Click
        * เมื่อผู้ใช้คลิกปุ่ม เพิ่ม Node โปรแกรมจะอ่านค่าสำหรับกำหนด Key ใน ThisForm.cKey แล้วเพิ่มขึ้นอีก 1
        tvwLast = 1
WITH ThisForm.Tree
                ThisForm.cKey = ThisForm.cKey+1    && เพิ่ม Key เข้าไปอีก 1
                nKey = ThisForm.cKey
                cRoot  = 'R'+ALLTRIM(STR(nKey))              && กำหนดชื่อที่จะใช้เป็น Key ซึ่งจะซ้ำกันไม่ได้
                cText = 'Item...('+ALLTRIM(STR(nKey))+')'    && ข้อความที่จะแสดงก่อนแก้ไข
                xNode = .Nodes.Add(,tvwLast,cRoot,cText)     && เพิ่ม Nodes เข้ากับ Tree View ในลำดับสุดท้าย
                * เมื่อโปรแกรมประมวลผลคำสั่งตามบรรทัดที่ผ่านมาแล้ว xNode จะเป็น Object ของ Tree View ซึ่ง
* เราสามารถที่จะดำเนินการโดยกำหนดพร็อปเพอร์ตี้ได้โดยตรง
                xNode.Selected = .T.  && กำหนดพร็อปเพอร์ตี้ Selected ให้เป็น .T. เป็นการขยาย TreeView ไปในตัว
                .StartLabelEdit              && กำหนดให้แก้ไขข้อความ
        ENDWITH

Control: 5 - btnChild
Events : Click
* เมื่อผู้ใช้คลิกปุ่ม เพิ่ม Child โปรแกรมจะตรวจสอบจำนวน Node ก่อนถ้าจำนวน Node ทั้งหมดใน TreeView
* เป็น 0 โปรแกรมจะไม่ยอมให้เพิ่ม Child
        WITH ThisForm.Tree
                IF .Nodes.Count <> 0
                        cRoot = .SelectedItem.Key
                        nIdx = .SelectedItem.Index          && เก็บค่า Index ของ Node ที่เลือกไว้ใน nIdx
                        nCnt = .Nodes(nIdx).Children+1      && นับจำนวน Child ใน Node ที่เลือกคลิก
* กำหนดชื่อคีย์ของ Child ซึ่งจะต้องไม่ซ้ำกัน
                cSub = 'S'+ALLTRIM(STR(nIdx))+ALLTRIM(STR(nCnt))
                        cText = 'Sub Item...('+ALLTRIM(STR(nCnt))+')'    && ข้อความที่จะแสดง
                        xNode = .Nodes.Add(cRoot,4,cSub,cText)                    && เพิ่ม Child ให้กับ Node ที่เลือก
                ELSE
                        =MESSAGEBOX('ยังไม่มีกลุ่มหลักให้เลือกเพิ่ม Nodes ก่อน',0+64+0,'Informarion')
                        RETURN
```

```
                ENDIF
                xNode.Selected = .T.    && เลือก Child ที่เพิ่มเข้าไปใหม่
                .StartLabelEdit                 && กำหนดให้แก้ไขข้อความของ Child ใหม่ที่เพิ่มเข้าไป
        ENDWITH

Control: 6 - btnDelete
Events : Click
        * เมื่อผู้ใช้คลิกปุ่ม ลบ Node โปรแกรมจะตรวจสอบก่อนว่ามีการคลิกเลือก Node แล้วหรือยัง ถ้ายังไม่คลิกเลือก
        * ให้แสดงข้อความเตือน แล้วย้อนกลับไปยังโปรแกรมตามเดิม
        WITH ThisForm.Tree
        IF .Nodes.Count = 0
                =MESSAGEBOX('ไม่พบข้อมูลที่ต้องการจะลบ',0+48+0,'Warning')
                RETURN
        ENDIF
        delKey = .SelectedItem.Key          && เก็บคีย์ที่จะลบไว้ในตัวแปร delKey
        IF !EMPTY(delKey)
                .Nodes.Remove(delKey) && ลบคีย์ตามชื่อที่เก็บไว้ใน delKey
                ThisForm.cKey = ThisForm.cKey-1     && เพิ่ม key ใน ThisForm.cKey อีก 1
        ENDIF
        ENDWITH

Control: 7 - btnExit
Events : Click
        * เมื่อคลิกเลือกปุ่ม ยกเลิกให้ลบฟอร์ม
        Release ThisForm
```

ส่งท้ายก่อนจบ

เห็นหรือยังละครับว่า Visual FoxPro ก็สามารถใช้งาน COM ที่เป็นมาพร้อมกับ Windows ได้อย่างไร้ปัญหา แถมจะง่ายกว่าหลาย ๆ โปรแกรมก็เป็นได้ ขอเพียงอย่างเดียวอย่าเพิ่งคิดท้อถอย เพราะปัญหาทุกปัญหามีทางออกเสมอ ถ้าเราพยายามอย่างเต็มที่และเต็มความสามารถ

บทความฉบับนี้ดูผิวเผินแล้วจะเยิ่นเย้อไปสำหรับผู้ที่ชำนาญ Visual FoxPro ชนิดหาตัวจับได้ แต่ที่ผ่านมาผู้เขียนพบว่าบางครั้งที่ผู้เขียนคิดว่าท่านผู้อ่านทราบกันดีแล้วไม่กล่าวถึง ปรากฏว่าผู้เขียนจะได้รับ e-mail สอบถามเพิ่มเติมเป็นประจำ ดังนั้นผู้เขียนจึงคิดว่าน่าจะบอกรายละเอียดให้ชัดเจนดีกว่าเพื่อให้ครอบคลุมถึงผู้อ่านหลาย ๆ กลุ่ม สำหรับผู้ที่ทราบพื้นฐานแล้วก็ถือว่าร่วมทบทวนความทรงจำหรือจะดูเฉพาะส่วนที่คิดว่าเป็นสาระสำคัญก็แล้วกัน

และบทความฉบับนี้ก็ต้องขอยุติเรื่องราวของ ActiveX ไว้เพียงเท่านี้ส่วนฉบับต่อไปจะกล่าวถึง ActiveX ตัวต่อไปขอโปรดติดตามต่อไปและถ้าต้องการติชมกรุณาส่ง e-mail ไปที่ yothinin@hotmail.com สำหรับที่ผ่านมาอาจจะตอบล่าช้าไปบ้างต้องขออภัยมา ณ ที่นี้ด้วย แต่ขอสัญญาว่าจะตอบให้ได้ทุกท่านนะครับ